University of Bahrain
College of Information Technology
Department of Computer Science
Summer Semester, 2008-2009
ITCS215 (Data Structures)

Mid-Term Exam

Date: 02/08/2008 Time: 08:00PM - 09:30PM

STUDENT NAME	
STUDENT ID #	
SECTION #	

NOTE: THERE ARE EIGHT **(8) PAGES** IN THIS TEST
ONLY ONE SOLUTION WILL BE CONSIDERED FOR EACH QUESTION

QUESTION #	MARKS		COMMENTS
1	12		
2	12		
3	12		
4	12		
5	12		
TOTAL	60		

www.uob-bh.com

(1) **Question 1 [12 Marks]**

Consider the following class definition:

```
class Person
{
    private:
          string name;
          char gender;
    public:
          Person (string personName, char personGender);
          string getName( );
          char getGender( );
          void print( );// Prints name and gender
};
```

(A) Write a class called ***Student*** which inherits all the properties of class ***Person*** with inheritance type as public. This new class will have the following additional members:
Data members: idNum (long), gpa (float).
Member Functions: set and get functions for both data members, print function to print all the attributes (including that of `Person`) and a suitable constructor function (with parameters).
Write only prototypes of all member functions in the class ***Student***.

(B) Write definitions (implementation) of the following member functions of class ***Student***:
constructor and `print`.

Question 2 [12 Marks]

Write a function **`isPalindrome`** that takes an object `L1` of type **`arrayListType`** as parameter. The function returns `true` if the object `L1` is palindrome, otherwise, it returns `false`. If `L1` has less than or equal to one element, then it's a palindrome.
Note: A list is palindrome if it reads the same forward and backward, such as the words "madam" or "radar".
Function prototype:

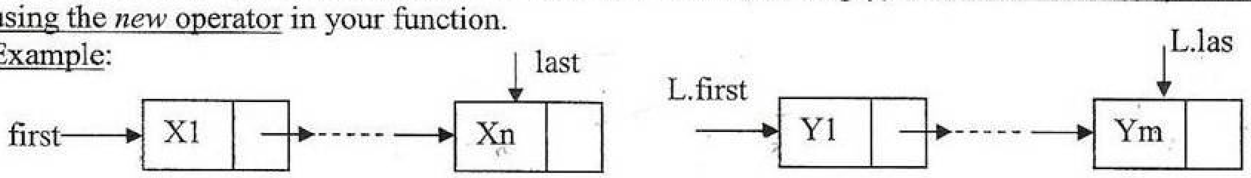
```
bool isPalindrome(arrayListType<Type>&  L1);
```

www.uob-bh.com

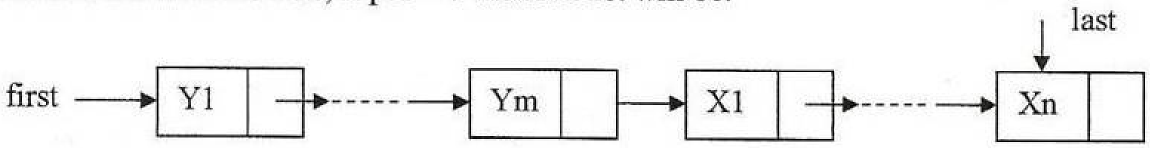
Question 3 [12 Marks]

Write a *member function* **`insertAt`** to be included in class **`linkedListType`** that accepts another **`linkedListType`** object **`L`** and an integer **`pos`** as parameters. The function connects all the nodes of list **`L`** *before* the position specified by **`pos`**. If **`pos`** is less than or equal 1, then connect the list at *front*. If **`pos`** is larger than the number of nodes in the list then connect list at the *end*. Assume that both lists have at least one node (not empty). Do NOT create any node using the *new* operator in your function.

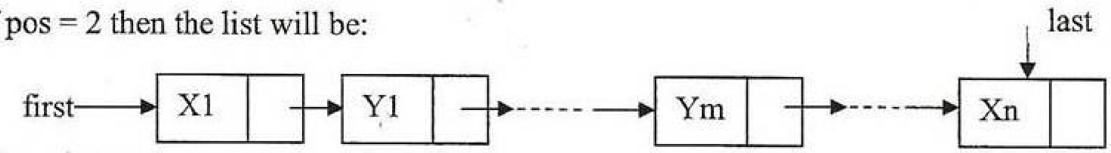
Example:



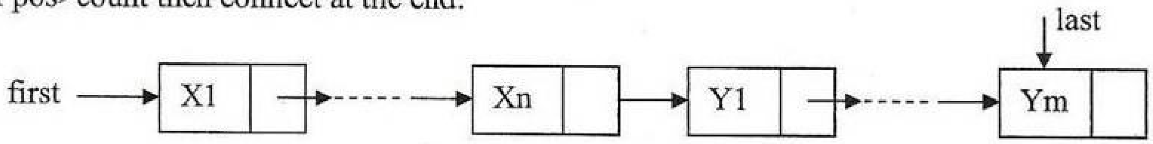
Given the above two lists, if pos = 1 then the list will be:



if pos = 2 then the list will be:



if pos>count then connect at the end:



Function prototype:

```
void insertAt (linkedListType<Type>& L, int pos)
```

www.uob-bh.com

Question 4 [12 Marks]
Write a *member function* called **`isReverseEqual`** to be included in class **`doublyLinkedList`**, that accepts another list `otherList` of type **`doublyLinkedList`** as parameter. The function returns `true` if the nodes of "this list" and `otherList` have same `info` but in reverse order, else it returns false.
Function prototype:
```
bool isReverseEqual(const doublyLinkedList<Type>& otherList);
```

Example: If the lists are as follows:

"this list":	5	10	15	18	30	35	4	50
otherList:	50	4	35	30	18	15	10	5

Then the function will return true.

Question 5 [12 Marks]
Using stack operations only, write a function **`rearrangeStack`** that takes a stack `st` as parameter and rearranges the stack into two parts. The first part (bottom) will have all odd numbers in the same order as they appear in the original stack `st` and the second part (top) will have all even numbers, also in the same order as in the original stack `st`. Use only common stack operations.
Function prototype:
```
void rearrangeStack(stackType<Type>& st);
```

www.uob-bh.com